OIPE JC101 MAY 0 2 2005 PATENT & TRADEMARK OFFICE

2126 AF 2126

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Application No.	:	09/431,449	Confirmation No.	:	8736
Applicant	:	GALLUSCIO et al.			
Filed	:	November 1, 1999			
TC/A.U.	:	2126			
Examiner	:	HOANG, Phuong N.			
Docket No.	:	6572-0014			
Customer No.	:	39207			

TRANSMITTAL LETTER

Mail Stop Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

Please find enclosed for filing:

- ✓ Appeal Brief (in triplicate)
- ✓ Fee: $500
- ✓ Other: Postcard Receipt
- ✓ Please charge any deficiencies or credit any overpayments to Deposit Acct. No. 50-2884.

Respectfully submitted,

Date: 4-29-05

[illegible]

Robert J. Sacco
Registration No. 35,667
Terry W. Forsythe
Registration No. 47,569
Sacco & Associates, P.A.
P.O. Box 30999
Palm Beach Gardens, FL 33420-0999

Tel: 561-626-2222

Certificate Under 37 C.F.R. 1.8(a)

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as First Class Mail in an envelope addressed to Mail Stop Appeal Brief – Patents, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on 4-29-05 Typed name of person signing this certificate: ROBERT J. SACCO

[illegible]

{00005862;}

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Application No.	:	09/431,449	Confirmation No.	:	8736
Applicant	:	GALLUSCIO et al.			
Filed	:	November 1, 1999			
TC/A.U.	:	2126			
Examiner	:	HOANG, Phuong N.			
Docket No.	:	6572-0014			
Customer No.	:	39207			

APPEAL BRIEF UNDER 37 C.F.R. § 41.31

Mail Stop Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

REAL PARTY IN INTEREST

The patent application to which the present appeal brief pertains, namely patent application no. 09/431,449, is assigned to Harris-Exigent, Inc. of Melbourne, Florida.

RELATED APPEALS AND INTERFERENCES

There are no related interferences, appeals or judicial proceedings known to Appellants, Appellants' legal representative, or the assignee which are related to this matter.

Certificate Under 37 C.F.R. 1.8(a)
I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as First Class Mail in an envelope addressed to Mail Stop Appeal Brief – Patents, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on 4-29-05 Typed name of person signing this certificate: ROBERT J. SACCO

[illegible]

STATUS OF CLAIMS

Claims 1-20 are pending in the application and stand rejected. Claims 1-20 are appealed.

STATUS OF AMENDMENTS

An amendment filed subsequent to the final office action was not entered. The claims presented herein are in the form in which they were presented to the Examiner prior to the final rejection.

SUMMARY OF THE CLAIMED SUBJECT MATTER

The present invention relates to a method and apparatus for high speed interprocess communications ("IPC"). In conventional IPC, multiple processes can communicate with one another via a shared region of random access memory ("RAM") to which each process can write data, and from which each process can read data. When communicating through the shared region of RAM, a first process functioning as the message source can write a message to the shared region of RAM. Subsequently, the second process, functioning as a message receiver, can read the written message from the shared region of RAM. Thus, at a minimum, two system calls are required to move n bytes of data from the first process to the second process through the shared region of RAM. Moreover, 2*n bytes of data will be stored in total – n bytes into the shared region of RAM, and n bytes into user memory space associated with the second process.

To overcome the excessive overhead associated with conventional IPC utilizing a shared region of RAM, the high speed IPC method and apparatus of the present invention avoids moving 2*n bytes of data by passing to the second process a memory offset which can be used by the second process to access the data. Importantly, the second process can manipulate and modify the data in place within the shared region of RAM. Accordingly, it is not required that the data be copied from the shared region of RAM to an alternate location while the manipulation takes place, thereby improving system efficiency.

Claim 1 recites a method for high speed interprocess communications comprising the step of attaching first and second processes (11, 12) to a message buffer (25) in a shared region (15) of random access memory (RAM) exclusive of operating system kernel space (14), each said process having a message list. p. 8, lines 15-28; p. 9, line 29 – p. 10, line 2; p. 10 lines 24-27. In addition, the claimed method recites accumulating message data (30) from the first process in a location in the message buffer (25), and adding to the message list of said second process a memory offset (29) corresponding to the location in the message buffer. p. 8, lines 26-28; p. 10, lines 10-15. Claim 1 further recites manipulating in the second process the accumulated data at said location corresponding to said offset. p. 12, lines 19-21. The accumulated message data is transferred from the first process to the second process with minimal data transfer overhead. p. 6, lines 1-3; p. 15, lines 11-13.

Claims 2 and 3 depend from claim 1. Claim 2 recites that the attaching step comprises the following steps: detecting a previously created shared region of RAM; if a shared region of RAM is not detected, creating and configuring a shared region of RAM for storing accumulated data; and, attaching the first and second processes to the shared region. p. 4, lines 6-10; p. 13, lines 1-21. Claim 3 recites that the message list is a message queue. p. 8, lines 25-29.

Claim 4 depends from claim 3, and recites that the adding step comprises the steps of retrieving a memory offset in the message buffer corresponding to the location of data accumulated by the first process, and inserting the memory offset in the message queue corresponding to the second process. p. 13, line 28 - p. 14, line 1; p. 14 lines 13-17. Claim 5 further recites that the inserting step comprises the step of atomically assigning the memory offset to an integer location in the message queue corresponding to the second process. p. 14, lines 18-19.

Claim 6 depends from claim 1, and recites that the processing step comprises the following steps: identifying a memory offset in the message list corresponding to the second process; processing in the second process message data stored at a location in the message buffer corresponding to the memory offset; and releasing the message buffer. p. 4, line 24 – p. 5, line 1; p. 14, line 22 - p. 15, line 3. Claim 19 depends from claim 1 and recites the step of locking the accumulated data to prevent the first process from accessing the accumulated data while the accumulated data is being manipulated. p. 13, lines 2-14.

Claim 7 recites a method for configuring high speed interprocess communications between first and second processes (11, 12). The method includes the steps of disposing a message buffer (25) in a shared region (15) of random access memory (RAM) shared between the first and second processes and accumulating message data (30) from said first process in a location in the message buffer. p. 9, lines 2-4; p. 13, line 9; p. 10, lines 10-11. Claim 7 also recites adding to a message list corresponding to the second process a memory offset corresponding to the location in the message buffer. p. 10, lines 11-15. In addition, claim 7 recites manipulating in the second process the accumulated message data stored in the message buffer at a location corresponding to the offset. p. 12, lines 19-21. The accumulated message data is transferred from said first process to said second process with minimal data transfer overhead. p. 6, lines 1-3; p. 15, lines 11-13.

Claim 8 depends from claim 7, and further recites that the disposing step comprises creating and configuring a message buffer in a shared region (15) of RAM exclusive of operating system kernel space (14). p. 8, lines 1-4; p. 13, lines 9-10. Claim 8 also recites creating a message list in the shared region for each process whereby the message list can store memory offsets (29) of message data (30) stored in the message buffer. p. 13, lines 20-23. Claim 9 recites that the message list is a message queue. p. 8, lines 25-26.

Claim 10 recites that the adding step comprises retrieving a memory offset in the message buffer, the memory offset corresponding to the location of the message data accumulated by the first process. p. 13, line 28 - p. 14, line 1. Claim 10 further recites inserting the memory offset in the message queue corresponding to the second process. p. 14 lines 15-17. Claim 11 recites that the inserting step comprises the step of atomically assigning the memory offset to an integer location in the message queue corresponding to the second process. p. 14, lines 18-19.

Claim 12 depends from claim 7 and recites the processing step comprises the following steps: identifying a memory offset in the message list corresponding to the second process; processing in the second process the accumulated message data at a location in the message buffer corresponding to the memory offset; and releasing said message buffer. p. 4, line 24 – p. 5, line 1; p. 14, line 22 - p. 15, line 3. Claim 20 depends from claim 7 and recites the step of locking the accumulated data to prevent the first process from accessing the accumulated data while the accumulated data is being manipulated. p. 13, lines 2-14.

Claims 13-18 recite a computer apparatus comprising means (e.g. RAM (15), CPU and program code) for performing the steps recited in claims 1-6. *See*, *e.g.*, p. 9, lines 14-23; p. 13, lines 4-7; p. 14, lines 5-12.

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Claims 7 and 9 -12 stand rejected under 35 U.S.C. §103(a) as being unpatentable over U.S. Patent No. 6,181,707 to Erickson et al. (hereinafter "Erickson") in view of U.S. Patent No. 5,504,901 Peterson (hereinafter "Peterson"). Claims 1-6, 8 and 13-18 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Erickson in view of Peterson, and in further view of U.S. Patent No. 6,148,377 to Carter et al. (hereinafter "Carter"). Finally, claims 19 and 20 were rejected under 35 U.S.C. §103(a) as being unpatentable over Erickson in view of Peterson, in view of Carter, and in further view of U.S. Patent No. 5,991,845 to Bohannon et al. (hereinafter "Bohannon").

ARGUMENT

The following claim groupings identify groups of claims which are different in scope and believed to be separately patentable. Accordingly, the claims of any particular group do not stand or fall with claims of any other group.

A. Claims 1 and 13

1. The cited references do not teach or suggest the recited first and second processes attached to a message buffer in a shared region of RAM wherein each said process has a message list.

In rejecting claims 1 and 13, the Examiner has primarily relied upon the Erickson reference. Erickson discloses a process which is entirely different than the claimed invention, however. Erickson discloses multiple matrix cards (2) and configuration cards (4) communicating via a TDM Bus. Col. 4, lines 53-59. Each of the cards has its own microprocessor, CDCC and dual-port RAM. Col. 6, lines 35-53. Notably, Erickson has utilized the same identifier (20) to identify dual-port RAM contained on different matrix cards. The Examiner may not have fully appreciated this nomenclature. Indeed, Erickson teaches that data is exchanged between the cards, col. 6, lines 45-53, but not that the cards share data at a single location as claimed by Appellants.

Claims 1 and 13 each recite, inter alia, attaching first and second processes to a message buffer in a shared region of random access memory (RAM), each of the processes having a message list. Accordingly, the first and second processes can pass memory offsets between each other to reference data in a common region of RAM, despite having different memory maps of the memory region. Erickson, Peterson and Carter each fail to teach or suggest the recited limitation. The Examiner has asserted that this limitation is taught by Erickson at col. 7, lines 4-55. Appellants respectfully disagree.

The RAM disclosed at col. 7, lines 4-55 is dual-port RAM 20, which the microprocessor 30 and CDCC 100 access from opposite sides. Col. 7, lines 12-15. Thus, in citing this portion of Erickson it appears that the Examiner equates Erickson's microprocessor 30 to the claimed first process and Erickson's CDCC 100 to the second process. Accordingly, to suggest the recited limitation, the microprocessor 30 and CDCC 100 must each have a message list. However, this is not the case. Erickson does not suggest that the microprocessor 30 has a message list.

To the extent that the Examiner may assert that the Word-In-Queue register 112 is equivalent to the message list of the first process, it should be noted that the Word-In-Queue register 112 is part of the CDCC100. Col. 7, line 46, *see also* FIG. 5A. Thus, although the microprocessor can provide an offset to the Word-In-Queue register of the CDCC to identify the last message of a word, there is no such register in the microprocessor for receiving an offset provided by the CDCC.

Moreover, the plain meaning of the terms process and microprocessor, and as such terms are generally understood to those skilled in the art, are not synonymous. A process is a series of actions, changes, or functions that bring about an end or result. The American Heritage Dictionary (2nd coll. ed., 1991). In contrast, a microprocessor is a semiconductor central processing unit usually contained on a single integrated circuit chip. *Id.* Although a microprocessor may execute program code to generate a process, Erickson provides no teaching or suggestion that the microprocessor 30 generates a process having a message list and that such process is attached to the shared region of RAM.

2. The cited references do not teach or suggest manipulating data in the second process at the location corresponding to the memory offset.

Claims 1 and 13 recite manipulating in the second process the data accumulated at the location corresponding to the offset. Erickson, Peterson and Carter each fail to teach or suggest the recited limitation. The Examiner acknowledges that Erickson does not teach this step, Final Office Action, p. 5, lines 11-13, but asserts that this limitation is taught by Peterson at col. 7 lines 42-45 and col. 8 lines 62-65. Appellants respectfully disagree. Nowhere in either of the cited passages does Peterson teach or suggest manipulating in the second process the accumulated data at the location in the message buffer corresponding to the memory offset. Instead, Peterson discloses using an offset pointer to access the data. Col. 7, lines 46-50. Accessing the data is distinct from manipulating the data at the location. To access data means to make use of the data. The American Heritage Dictionary. Thus, data can be accessed by a particular process and transferred to a portion of memory where that process is executing. In contrast, the claimed step manipulates the data at the location corresponding to the offset rather than transferring the data to another portion of memory.

"To establish prima facie obviousness of a claimed invention, all the claim limitations must be taught or suggested by the prior art." MPEP § 2143.03, *citing In re Royka*, 490 F.2d 981, 985 (CCPA 1970); *see also CFMT, Inc. v. YieldUp Int'l Corp.*, 349 F.3d 1333, 1342 (Fed. Cir. 2003). Because neither Erickson nor Peterson teach or suggest the recited first and second processes each having a message list, or the recited step of manipulating data with the second process at the location corresponding to a memory offset, the Examiner has not established prima facie obviousness. Accordingly, the rejection of claims 1 and 13 is improper and must be overturned. *In re Ochiai*, 71 F.3d 1565, 1569 (Fed. Cir. 1995).

Claims 3, 4, 6, 15, 16 and 18-20 are believed allowable at least based on their dependence on an allowable base claim.

B. Claims 2 and 14

Claims 2 and 14 each recite detecting a previously created shared region of RAM and, if the shared region is not detected, creating and configuring a shared region of RAM for accumulated data. The first and second processes are then attached to the shared region. Erickson, Peterson and Carter each fail to teach or suggest the recited limitation. The Examiner asserts that these limitations are taught by Erickson at col. 7, lines 15-30. Appellants disagree. Erickson wholly fails to disclose detecting a previously created shared region of RAM and creating and configuring one if not detected. At most, the CDCC indicates the availability of a transmit queue. However, there is no teaching or suggestion of creating the transmit queue if it is unavailable.

C. Claims 5, 11 and 17

Claims 5, 11 and 17 each recite atomically assigning the memory offset to an integer location in the message queue corresponding to the second process. Accordingly, an entire integer memory offset can be written to the message queue using a single instruction. In contrast, the traditional method of copying a data message using a memory copy requires an assignment for each byte in the data message. See Appellants' specification, p. 11, line 25 – p. 12, line 5.

None of the cited references teach or suggest the recited limitation. The Examiner asserts that Erickson discloses atomic assignment of the memory offset at col. 7, lines 35-55. Appellants disagree. Nowhere in the cited portion does Erickson teach or suggest atomically assigning the memory offset. Indeed, Erickson does not even disclose that it is desirable to write the memory offset in a single instruction.

D. Claim 7

1. The cited references do not teach or suggest the recited first and second processes which share a region of RAM.

Claim 7 recites, inter alia, disposing a message in a shared region of random access memory (RAM) shared between first and second processes. Accordingly, the first and second processes can pass memory offsets between each other to reference data in a common region of RAM, despite having different memory maps of the memory region. None of the cited references teach or suggest the recited limitation. The Examiner has asserted that this limitation is taught by Erickson at col. 7, lines 4-55. Appellants respectfully disagree.

As previously noted, the terms process and microprocessor are not synonymous. A process is a series of actions, changes, or functions that bring about an end or result. The American Heritage Dictionary (2nd coll. ed., 1991). In contrast, a microprocessor is a semiconductor central processing unit usually contained on a single integrated circuit chip. *Id.* Although a microprocessor may execute program code to generate a process, Erickson provides no teaching or suggestion that the microprocessor 30 generates a process having a message list and attaches such process to the shared region of RAM.

2. <u>The cited references do not teach or suggest manipulating data in the second process at the location corresponding to the memory offset.</u>

Claim 7 also recites manipulating in the second process the data accumulated at the location corresponding to the offset. None of the cited references teach or suggest the recited limitation. The Examiner acknowledges that Erickson does not teach this step, Final Office Action, p. 5, lines 11-13, but asserts that this limitation is taught by Peterson at col. 7 lines 42-45 and col. 8 lines 62-65. Appellants respectfully disagree. Nowhere in either of the cited passages does Peterson teach or suggest manipulating in the second process the accumulated data at the location in the message buffer corresponding to the memory offset. Instead, Peterson discloses using an offset pointer to access the data. Col. 7, lines 46-50. Accessing the data is distinct from manipulating the data at the location. To access data means to make use of the data. The American Heritage Dictionary. Thus, data can be accessed by a particular process and transferred to a portion of memory where that process is executing. In contrast, the claimed step manipulates the data at the location corresponding to the offset rather than transferring the data to another portion of memory.

Because neither Erickson nor Peterson teach or suggest the recited first and second processes, or the recited step of manipulating data with the second process at the location corresponding a memory offset, the Examiner has not established prima facie obviousness. Appellants therefore respectfully request that the rejection of claim 7 be overturned. *See In re Ochiai*, 71 F.3d at 1569.

Claims 9, 10 and 12 are believed allowable at least based on their dependence on an allowable base claim.

E. <u>Claim 8</u>

Claim 8 recites, inter alia, the step of creating a message list in the shared region of RAM for each process. Erickson, Peterson and Carter each fail to teach or suggest the recited limitation. The Examiner has asserted that this limitation is taught by Erickson at col. 7, lines 1-55. Appellants respectfully disagree.

The RAM disclosed at col. 7, lines 1-55 is dual-port RAM 20, which the microprocessor 30 and CDCC 100 access from opposite sides. Col. 7, lines 12-15. Thus, it appears that the Examiner equates Erickson's microprocessor 30 to the claimed first process and Erickson's CDCC 100 to the second process. Accordingly, to suggest the recited limitation, the microprocessor 30 and CDCC 100 each must have a message list created in the shared region of RAM. However, this is not the case.

Erickson does not suggest that the microprocessor 30 even has a message list. To the extent that the Examiner may assert that the Word-In-Queue register 112 is equivalent to the message list of the first process, it should be noted that the Word-In-Queue register 112 is part of the CDCC 100. Col. 7, line 46, *see also* FIG. 5A. Moreover, Erickson provides no teaching or suggestion that the Word-In-Queue register is created in the shared region of RAM. Indeed, as shown in FIG. 5A, the Word-In-Queue Register 112 is not contained in the RAM, but instead is an entirely different structure contained in the CDCC 100.

CONCLUSION

Appellants' invention relates to a method and apparatus for high speed interprocess communications ("IPC"). The high speed IPC method and apparatus of the present invention improves communication efficiency by passing data from a first process to a second process using a memory offset. The memory offset is used by the second process to access the data without requiring the data to be moved. Thus, the process eliminates the need for copying the data from the shared region of RAM to an alternate location while data manipulation takes place.

In the Examiner's rejection of claims 1-20, the cited references have been misguidedly applied. The cited references fail to render Appellants' invention obvious. Accordingly, Appellants submit that claims 1-20 define a patentably distinguishable invention over the prior art made of record, and a Notice of Allowance for claims 1-20 is accordingly and courteously solicited.

Respectfully submitted,

Date: 4-29-05

[illegible]

Robert J. Sacco
Registration No. 35,667
Terry W. Forsythe
Registration No. 47,569
Sacco & Associates, P.A.
P.O. Box 30999
Palm Beach Gardens, FL 33420-0999
Tel: 561-626-2222

APPENDIX

CLAIMS

1. A method for high speed interprocess communications comprising the steps of:

attaching first and second processes to a message buffer in a shared region of random access memory (RAM) exclusive of operating system kernel space, each said process having a message list;

accumulating message data from said first process in a location in said message buffer;

adding to said message list of said second process a memory offset corresponding to said location in said message buffer; and,

manipulating in said second process said accumulated data at said location corresponding to said offset,

whereby said accumulated message data is transferred from said first process to said second process with minimal data transfer overhead.

2. The method according to claim 1, wherein the attaching step comprises the steps of:

detecting a previously created shared region of RAM;

if a shared region of RAM is not detected, creating and configuring a shared region of RAM for storing accumulated data; and,

attaching said first and second processes to said shared region.

3. The method according to claim 1, wherein said message list is a message queue.

4. The method according to claim 3, wherein the adding step comprises the steps of:

retrieving a memory offset in said message buffer corresponding to said location of data accumulated by said first process; and, inserting said memory offset in said message queue corresponding to said second process.

5. The method according to claim 4, wherein the inserting step comprises the step of atomically assigning said memory offset to an integer location in said message queue corresponding to said second process.

6. The method according to claim 1, wherein the processing step comprises the steps of:

identifying a memory offset in said message list corresponding to said second process;

processing in said second process message data stored at a location in said message buffer corresponding to said memory offset; and,

releasing said message buffer.

7. A method for configuring high speed interprocess communications between first and second processes comprising the steps of:

disposing a message buffer in a shared region of random access memory (RAM) shared between said first and second processes;

accumulating message data from said first process in a location in said message buffer;

adding to a message list corresponding to said second process a memory offset corresponding to said location in said message buffer; and,

manipulating in said second process said accumulated message data stored in said message buffer at a location corresponding to said offset,

whereby said accumulated message data is transferred from said first process to said second process with minimal data transfer overhead.

8. The method according to claim 7, wherein the disposing step comprises the steps of:

creating and configuring a message buffer in a shared region of RAM exclusive of operating system kernel space; and,

creating a message list in said shared region for each said process,

whereby said message list can store memory offsets of message data stored in said message buffer.

9. The method according to claim 7, wherein said message list is a message queue.

10. The method according to claim 9, wherein the adding step comprises the steps of:

retrieving a memory offset in said message buffer, said memory offset corresponding to said location of said message data accumulated by said first process; and,

inserting said memory offset in said message queue corresponding to said second process.

11. The method according to claim 10, wherein the inserting step comprises the step of atomically assigning said memory offset to an integer location in said message queue corresponding to said second process.

12. The method according to claim 7, wherein the processing step comprises the steps of:

identifying a memory offset in said message list corresponding to said second process;

processing in said second process said accumulated message data at a location in said message buffer corresponding to said memory offset; and,

releasing said message buffer.

13. A computer apparatus programmed with a set of instructions stored in a fixed medium for high speed interprocess communications, said programmed computer apparatus comprising:

means for attaching first and second processes to a message buffer in a shared region of random access memory (RAM) exclusive of operating system kernel space, each said process having a message list;

means for accumulating message data from said first process in a location in said message buffer;

means for adding to said message list of said second process a memory offset corresponding to said location in said message buffer; and,

means for manipulating in said second process said accumulated data at said location corresponding to said offset.

14. The computer apparatus of claim 13, wherein the attaching means comprises:

means for detecting a previously created shared region of RAM; and,

means for creating and configuring a shared region in RAM for storing accumulated data if a previously created shared region of RAM is not detected by said detecting means.

15. The computer apparatus according to claim 13, wherein said message list is a message queue.

16. The computer apparatus according to claim 15, wherein the adding means comprises:

means for retrieving a memory offset in said message buffer corresponding to said location of data accumulated by said first process; and,

means for inserting said memory offset in said message queue corresponding to said second process.

17. The computer apparatus according to claim 16, wherein the inserting means comprises means for atomically assigning said memory offset to an integer location in said message queue corresponding to said second process.

18. The computer apparatus according to claim 13, wherein the processing means comprises:

means for identifying a memory offset in said message list corresponding to said second process;

means for using in said second process message data at a location in said message buffer corresponding to said memory offset; and,

means for releasing said message buffer.

19. The method according to claim 1, further comprising the step of locking said accumulated data to prevent said first process from accessing said accumulated data while said accumulated data is being manipulated.

20. The method according to claim 13, wherein said accumulated data is locked to prevent said first process from accessing said accumulated data while said accumulated data is being manipulated.